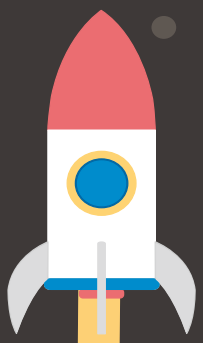
Modul Praktikum
Teori Bahasa & Otomata



Language

Finite Automata

Context Free Grammar

Pushdown Automata

Regular Expression

Chomsky Normal Form

Oleh:
Sri Handayaningsih
Andri Pranolo

Sejarah Revisi Modul Praktikum Teori Bahasa Automata

No	Tahun Revisi	Revisi
1	2017	Penggunaan tool JFLAP

Detail Sejarah Revisi Modul Praktikum Teori Bahasa Automata

Modul Revisi ke 1 tahun 2016	Modul Revisi ke 2 tahun 2017
Pertemuan I. Pengolahan String	Pertemuan I. Pengolahan String
Pertemuan II. Pengolahan String II	Pertemuan II. Pengolahan String II
Pertemuan III. Operasi pada Bahasa	Pertemuan III. Operasi pada Bahasa
Pertemuan IV. Finite State Authomata (FSA)	**Pertemuan IV. Finite State Automata (FSA) dengan JFLAP**
Pertemuan V. Deterministic Finite Automata	Pertemuan V. NonDeterministic Finite Automat.
Pertemuan VI. NonDeterministic Finite Automata	Pertemuan VI. Deterministic Finite Automata
Pertemuan VII. NFA dengan transisi Epsilon	**Pertemuan VII. Pushdown Automata**
Pertemuan VIII. Chomsky Normal Form	Pertemuan VIII. Chomsky Normal Form
Pertemuan IX. Penghilangan Rekursif Kiri	Pertemuan IX. Penghilangan Rekursif Kiri
Pertemuan X. Normal Greibach	Pertemuan X. Normal Greibac.

Yogyakarta, Agustus 2017

Penulis

Daftar Isi

Pengolahan String

Pertemuan	:	I
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* untuk pengolahan string dengan menggunakan *visual programming* 2. Mahasiswa mampu memahami algoritam dari pengolahan string dalam hal ini panjang string, *Reverse* dan *Concatenation*
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming* 2. Mahasiswa mampu membuat fungsi pengolahan string untuk menghitung panjang string, *reverse*, dan *concatenation*

A. Dasar Teori

Terminologi dasar yang penting dalam memahami teori bahasa adalah alphabet, penyambungan (*Contatenation*) dan string pada alphabet V. alphabet digunakan untuk membentuk kata-kata di bahasa. Pada beberapa buku alphabet dilambangkan dengan ∑.

Kumpulan alphabet atau symbol disebut string. Ada banyak operasi pengolahan yang bisa dilakukan pada string yaitu *contetenation*, panjang string dan pembalikan (*reverse*).

1. *Concatenation* : penyambungan 2 karakter atau lebih membentuk suatu barisan karakter.
2. Panjang string : proses penghitungan jumlah karakter yang dimuat dalam suatu string
3. *Reverse* : pembalikan string

Misalnya u = abbbba

v = bbbbba

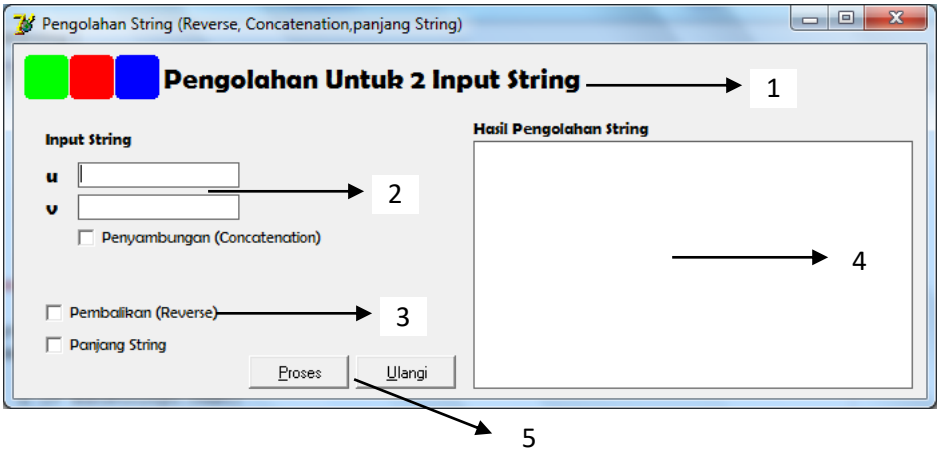
uv = abbbbabbbbba (penyambungan),

|uv| = |u| + |v| = 6 + 6 = 12 (panjang string)

$(uv)^R = (abbbbabbbbba)^R = abbbbbabbbba$

B. Langkah Praktikum

1. Buka program visual programming (dalam modul ini menggunakan Borland Delphi 7).
2. Buat Desain *form* seperti gambar 1.1



Gambar 1.1 . Desain *form* pengolahan string

No	Nama variable	Properties
1	*Label*	Posisi *Tab standard* *Caption* : diubah sesuai kebutuhan
2	*Edit*	Posisi *tab standard* *Name* : sesuaikan dengan input datanya
3	*Checkbox*	Posisi *tab standard* *Caption* : ubah sesuai dengan permintaan
4	*Memo*	Posisi *tab standard* *Name* : sesuaikan dengan kegunaan (missal memohasil) *Lines* : hapus semua teks yang ada di dalamnya
5	*Button*	Posisi *Tab standard* *Caption* : disesuai dengan kebutuhan *Name* : sesuai dengan nama caption

3. Jika desain *form* sudah selesai dilakukan masukkan coding di bawah ini :

Prosedur untuk menghitung panjang string

```
procedure TfrmUtama.Panjangstring;
var
  i: Integer;
begin
  u := edtstringu.Text;
  v := edtstringv.Text;

  if chkpenyambungan.Checked then
  begin
    mmohasil.Lines.Add('|uv| = |u| + |v| = '+IntToStr(Length(uv)))
  end
  else begin
    mmohasil.Lines.Add('|u| = '+IntToStr(Length(u)));
    mmohasil.Lines.Add('|v| = '+IntToStr(Length(v)));
  end;
end;
```

Prosedur untuk proses pembalikan pada string

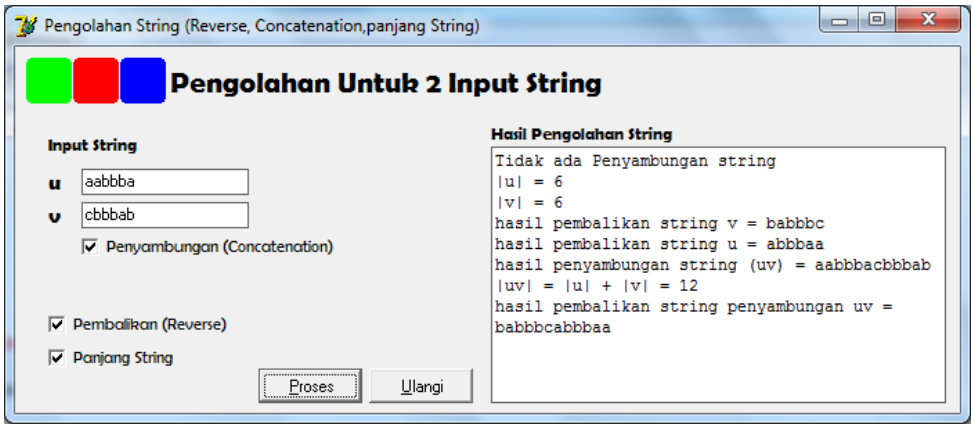
```
procedure TfrmUtama.Pembalikan;
var
  panjangu, panjangv: Integer;
  i,j, total: Integer;
begin
  u := edtstringu.Text;
  v := edtstringv.Text;

  panjangv := length(v);
  panjangu := length(u);
  total := Length(uv);
  if chkpenyambungan.Checked then
  begin
    edttampung.Text := '';
    for i := 0 to Length(uv) do
    begin
      edttampung.Text := edttampung.Text+''+uv[total - i];
    end;
    mmohasil.Lines.Add('hasil pembalikan string penyambungan uv = '+
edttampung.Text);
```

```
  end
  else begin
    edttampung.Text :='';
    for i:= 0 to Length(v) do
    begin
      edttampung.Text := edttampung.Text+''+v[panjangv - i];
    end;
    mmohasil.Lines.Add('hasil pembalikan string v = '+
edttampung.Text);

    edttampung.Text :='';
    for j:= 0 to Length(u) do
    begin
      edttampung.Text := edttampung.Text+''+u[panjangu - j];
    end;
    mmohasil.Lines.Add('hasil pembalikan string u = '+
edttampung.Text);
  end;
end;
```

Prosedur Penyambungan

```
procedure TfrmUtama.Penyambungan;
begin
  u := edtstringu.Text;
  v := edtstringv.Text;

  if chkpenyambungan.Checked then
  begin
    uv := u+''+v;
    mmohasil.Lines.Add('hasil penyambungan string (uv) =
'+uv);
  end
  else begin
    mmohasil.Lines.Add('Tidak ada Penyambungan string')
  end;
end;
```

4. Tekan F9 atau tombol run untuk menjalankan program
5. Coba masukkan 2 buah string dan lihat hasilnya cocokkan dengan hasil manualnya (contoh di bawah ini)



Gambar 1.2. *form* pengolahan *string* dengan contoh *output*

C. Tugas Praktikum

Buat program untuk melakukan pembalikan dan untuk menghitung panjang 3 buah string dengan ketentuan sebagai berikut u = aaaabbbbb , v = bbbbcccca , w = ccccbbbbaaa

Nilai	Yogyakarta, ……. Paraf asisten <……>
Jawaban Postest	

Pengolahan String 2

Pertemuan	:	II
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* untuk pengolahan string dengan menggunakan *visual programming* 2. Mahasiswa mampu memahami algoritma dari pengolahan string dalam hal ini *Prefix*, *Suffix*, *Star Clouser* dan *Positif Clouser*
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming* 2. Mahasiswa mampu membuat fungsi pengolahan string untuk *Prefix*, *Suffix*, *Star Clouser* dan *Positif Clouser*

A. Dasar Teori

Selain operasi penyambungan dan pembalikan ada beberapa operasi dasar pada string diantaranya *Prefix, suffix, star clouser* dan *positif clouser*.

Prefix string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling belakang dari string *w* tersebut.

Contoh : *abc*, *ab*, *a*, dan e adalah semua Prefix(*x*)

Postfix (atau *Suffix*) string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling depan dari string *w* tersebut.

Contoh : *abc*, *bc*, *c*, dan e adalah semua *Postfix(x)*.

Bahasa Universal

Jika $\sum$ adalah alphabet, kita menggunakan $\sum^*$ untuk menotasikan himpunan string(bahasa universal) yang dihasilkan oleh penggabungan nol atau lebih

symbol $\sum$. $\sum^*$ selalu mengandung λ agar bisa mengeluarkan string yang kosong. Sedangkan $\sum^+$ tidak mengandung string kosong λ atau $\sum^+ = \sum^* - \lambda$.

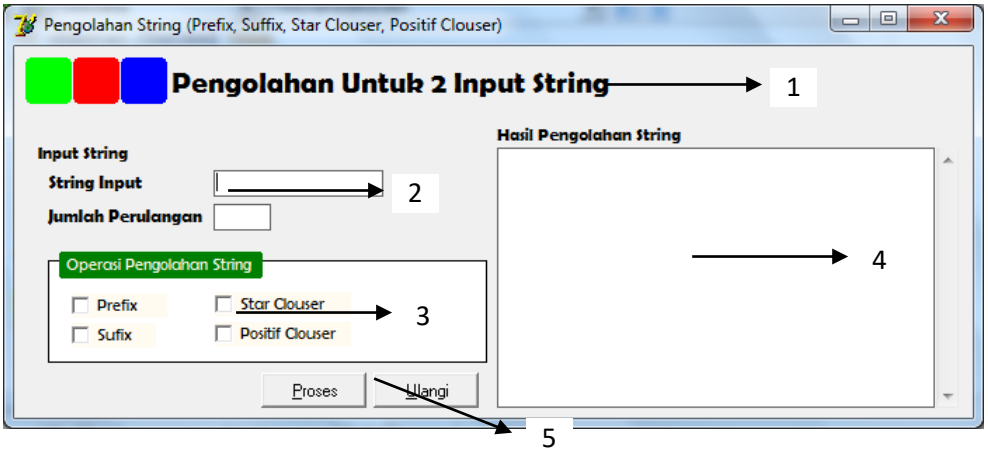
Contoh

$\sum^* = \{ \lambda, a, aa, aaa, aaaa, \ldots\ldots \}$

$\sum^+ = \{ a, aa, aaa, aaaa, \ldots\ldots \}$

B. Petunjuk Praktikum

1. Buka program visual programming (dalam modul ini menggunakan Borland Delphi 7).
2. Buat Desain *form* seperti gambar 2.1



Gambar 2.1. Desain form pengolahan string

No	Nama variable	Properties
1	*Label*	Posisi *Tab standard* *Caption* : diubah sesuai kebutuhan
2	*Edit*	Posisi *tab standard* *Name* : sesuaikan dengan input datanya
3	*Checkbox*	Posisi *tab standard* *Caption* : ubah sesuai dengan permintaan
4	*Memo*	Posisi *tab standard* *Name* : sesuaikan dengan kegunaan (missal memohasil) *Lines* : hapus semua teks yang ada di dalamnya
5	*Button*	Posisi *Tab standard* *Caption* : disesuai dengan kebutuhan *Name* : sesuai dengan nama caption

3. Jika desain *form* sudah selesai dilakukan masukkan coding di bawah ini :

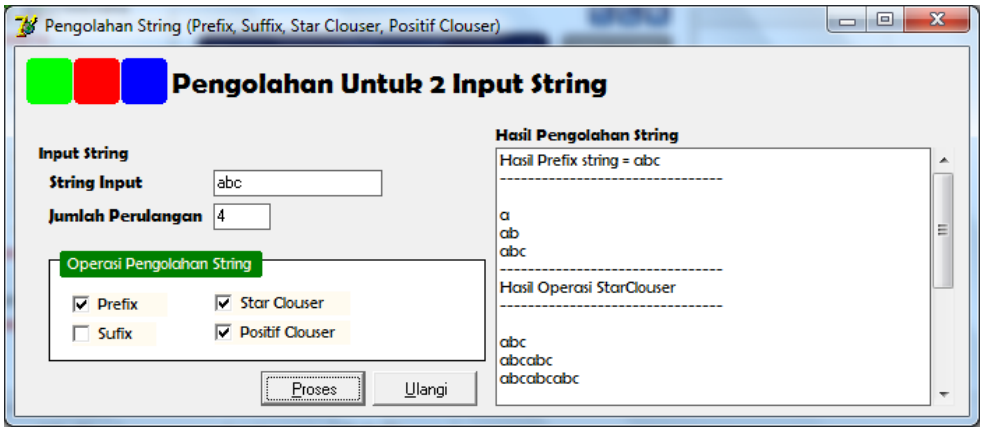
Prosedur untuk membuat *Prefix* dari string

```
procedure TfrmUtama.prefixmethod(karakter : string);
var
  i: Integer;
begin
  edttemplate.Text :='';
  mmohasil.Lines.Add('Hasil Prefix string = '+karakter);
  mmohasil.Lines.Add('--------------------------------');
  for i := 0 to Length(karakter) do
  begin
    edttemplate.Text :=edttemplate.Text+''+karakter[i];
    mmohasil.Lines.Add(edttemplate.Text)
  end;
  mmohasil.Lines.Add('--------------------------------');
end;
```

Prosedur membuat *starclouser* dari string

```
procedure TfrmUtama.starmethod(karakter : string;ulang : integer);
var
  i: Integer;
begin
  edttemplate.Text :='';
  mmohasil.Lines.Add('Hasil Operasi StarClouser ');
  mmohasil.Lines.Add('--------------------------------');
  mmohasil.Lines.Add('');
  for i := 0 to ulang-1 do
  begin
    edttemplate.Text := edttemplate.Text+''+karakter;
    mmohasil.Lines.Add(edttemplate.Text);
  end;
  mmohasil.Lines.Add('--------------------------------');
end;
```

4. Tekan F9 atau tombol run untuk menjalankan program
5. Coba masukkan sebuah string dan lihat hasilnya cocokkan dengan hasil manualnya (gambar 2.2)



Gambar 2.2. form program ketika sudah di *run*

C. Tugas Praktikum

Kembangkan program di atas, buat fungsi *Suffix* dan *Positif clouser*. Tentukan hasilnya untuk string :

1. baababb
2. aaaaabbbab
3. bababababba
4. acacccaa

(fungsi dan hasil dari pengolahan string di tuliskan di modul)

Nilai	Yogyakarta, …… Paraf asisten <……>
Jawaban Postest	

Operasi Bahasa

Pertemuan	:	III
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* untuk pengolahan string dengan menggunakan *visual programming* 2. Mahasiswa mampu memahami algoritma dari operasi 2 buah bahasa. Dalam hal ini *Union ,intersection, difference, concatenation, reverse*
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming* 2. Mahasiswa mampu membuat fungsi pengolahan bahasa *Union ,intersection, difference, concatenation, reverse*

A. Dasar Teori

Bahasa adalah subset dari $\sum^*$

Contoh: $\Sigma = \{a, b\}$

$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$

Bahasa: $\{\lambda\}$

$\{a, aa, aab\}$

$\{\lambda, abba, baba, aa, ab, aaaaaa\}$

Bahasa Tidak Terbatas $L = \{a^n b^n : n \geq 0\}$

λ
ab
$aabb$
$aaaaabbbbb$
$\in L$

$abb \notin L$

Operasi dalam bahasa

Operasi-operasi pada bahasa tidak jauh berbeda dengan operasi-operasi pada string. Ada *Reverse, concatenation* dan lain sebagainya. Selain itu juga sama dengan operasi pada himpunan *Union, Intersection danDiffrence*

Menggunakan operasi himpunan : $\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$

$\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$

$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$

Complement: $\overline{L} = \Sigma^* - L$ $\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$

Reverse

Definisi: $L^R = \{w^R : w \in L\}$

Contoh: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

$L = \{a^n b^n : n \geq 0\}$

$L^R = \{b^n a^n : n \geq 0\}$

Concatenation

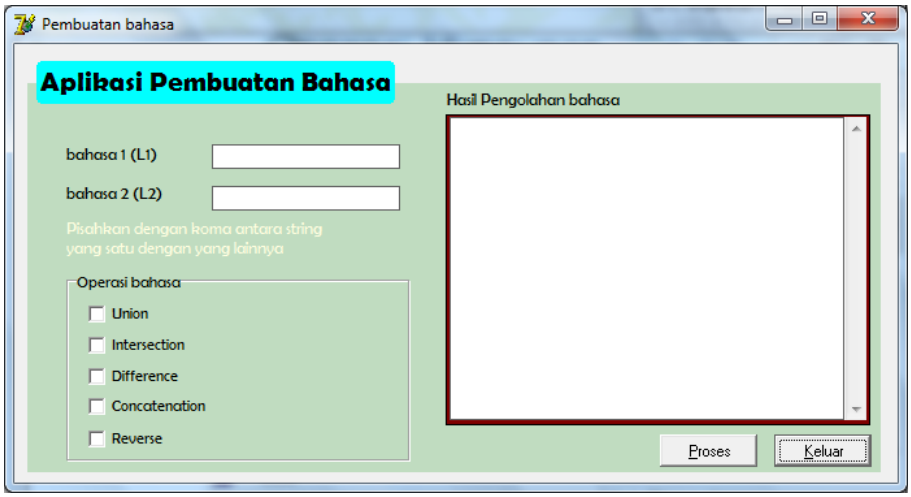
Definisi : $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

Contoh $\{a, ab, ba\}\{b, aa\}$

$= \{ab, aaa, abb, abaa, bab, baaa\}$

B. Petunjuk Praktikum

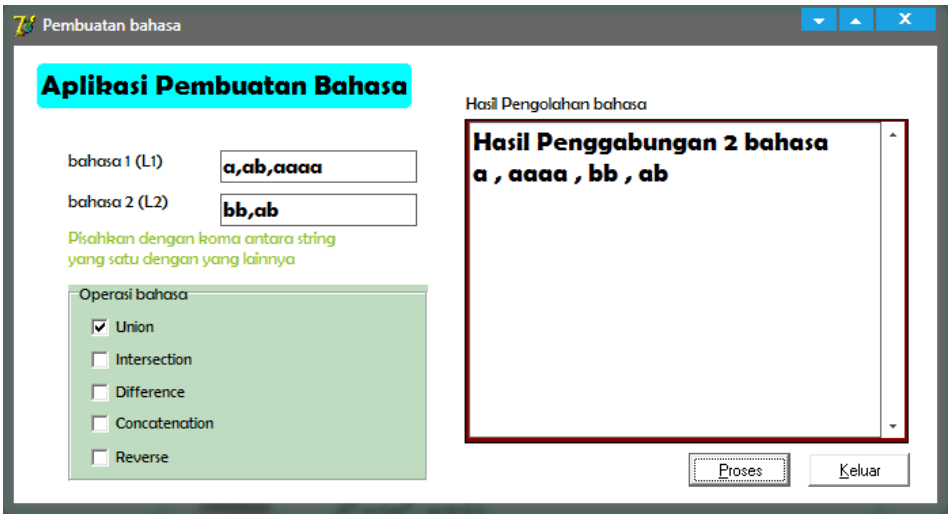
1. Buka program visual programming(dalam modul menggunakan Borland Delphi 7)
2. Desain form seperti di bawah ini (gambar 3.1)



Gambar 3.1. desain form aplikasi pembuatan bahasa

No	Nama variable	Properties
1	*Label*	Posisi *Tab standard* *Caption* : diubah sesuai kebutuhan
2	*Edit*	Posisi *tab standard* *Name* : sesuaikan dengan input datanya
3	*Checkbox*	Posisi *tab standard* *Caption* : ubah sesuai dengan permintaan
4	*Memo*	Posisi *tab standard* *Name* : sesuaikan dengan kegunaan (missal memohasil) *Lines* : hapus semua teks yang ada di dalamnya
5	*Button*	Posisi *Tab standard* *Caption* : disesuai dengan kebutuhan *Name* : sesuai dengan nama caption

3. Buat fungsi untuk proses *union* dan *intersection*
4. Tekan F9 atau tombol run untuk menjalankan program
5. Coba masukkan sebuah string dan lihat hasilnya cocokkan dengan hasil manualnya (contoh di bawah ini)



Gambar 3.2. contoh hasil pengolahan pada bahasa (operasi penggabungan)

Prosedur lengkap proses union (penggabugan 2 bahasa)

```
procedure Tfrmutama.memecahkanbahasa(kata : string);
var
  j: Integer;
  i: Integer;
  stringdata: string;
begin
j := 0;
 for i := 1 to Length(kata)+1 do
 begin
    if (kata[i] <> ',') and (kata[i] <> '') then
    begin
      stringdata := stringdata+''+kata[i];
    end
    else begin
      lstdata.Items.Strings[j] := stringdata;
      Inc(j);
      stringdata:='';
    end;
 end;
end;
```

```
procedure Tfrmutama.Union;
var
  k: Integer;
  j: Integer;
  hasilgabung: string;
  stringdata: string;
  i: Integer;
begin

  for i := lstdata.Items.Count - 1 downto 0 do
  begin
    if lstdata1.Items.IndexOf(lstdata.Items[i]) >= 0 then
      lstdata.Items.Delete(i);
  end;

  hasilgabung :=lstdata.Items.Strings[0];;
  mmohasil.Lines.Add('Hasil Penggabungan 2 bahasa');
  for j := 1 to lstdata.Items.Count - 1 do
  begin
    if lstdata.Items.Strings[j] <> '' then
      hasilgabung := hasilgabung+' , '+lstdata.Items.Strings[j];
  end;

  for k := 0 to lstdata1.Items.Count - 1 do
  begin
    hasilgabung := hasilgabung+' , '+lstdata1.Items.Strings[k];
  end;

  mmohasil.Lines.Add(hasilgabung);
  hasilgabung:=''
end;
```

C. Tugas Praktikum

Dari program di atas tambahkan fungsi untuk proses *Concatenation*. Cek hasilnya untuk bahasa di bawah ini:

1. L1 = {ab,bba} dan L2 = {bb,aba}
2. L1 = {ac, bc} dan L3 = {bc, ab, ac}

Nilai	Yogyakarta, …… Paraf asisten <……>
Jawaban Postest	

Finite State Automata (FSA) dengan JFLAP

Pertemuan	:	IV
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu memahami *Finite State Automata* dalam JFLAP 2. Mahasiswa mampu memahami algoritma *Finite State Automata*
Indikator	:	1. Mahasiswa mampu membuat simulasi FSA dengan JFLAP 2. Mahasiswa mampu menjelaskan hasil simulasi FSA dengan JFLAP

A. Dasar Teori

Finite automata adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana (bahasa reguler) dan dapat diimplementasikan secara nyata dimana sistem dapat berada disalah satu dari sejumlah berhingga konfigurasi internal disebut state. State sistem merupakan ringkasan informasi yang berkaitan dengan masukan-masukan sebelumnya yang diperlukan untuk menentukan perilaku sistem pada masukan-masukan berikutnya.

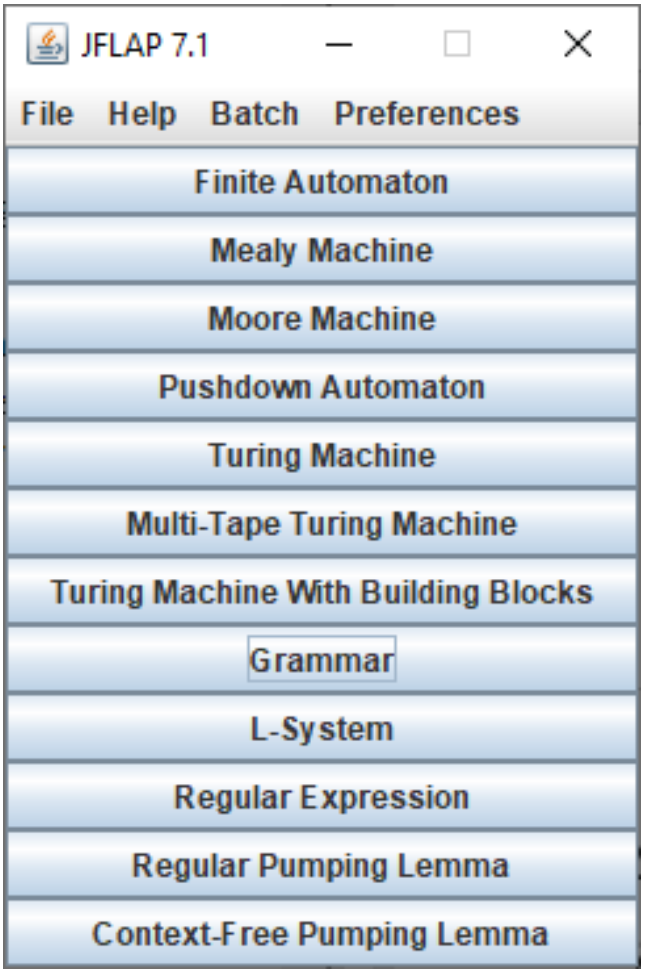
Finite Automata menggunakan prosedur yang saat diberikan masukan "string berhingga" akan berhenti. *Finite Automata* menyatakan "ya" dengan sejumlah berhingga komputasi jika string tersebut merupakan elemen bahasa sehingga lebih berfokus pada pengenalan dimana bila diberikan suatu program (string) akan menyatakan apakah string tersebut termasuk di bahasa atau tidak.

JFLAP mendepfinsikan sebuah a Finite State Automata (FA) M sebagai *quin* tupel, M = (Q, Σ, δ, qs/S, F) dimana

1. Q = adalah sebuah himpunan finite states {qi | I merupakan non-negatif integer}
2. ∑ = himpunan symbol *input* berupa alphabet
3. δ = fungsi transisi, δ : D → 2Q dimana D adalah sub himpunan finite state dari Q × Σ*
4. qs = merupakan anggota himpunan Q, menunjukan initial state.
5. F = adalah sub himpunan dari Q, merupakan state-state akhir.

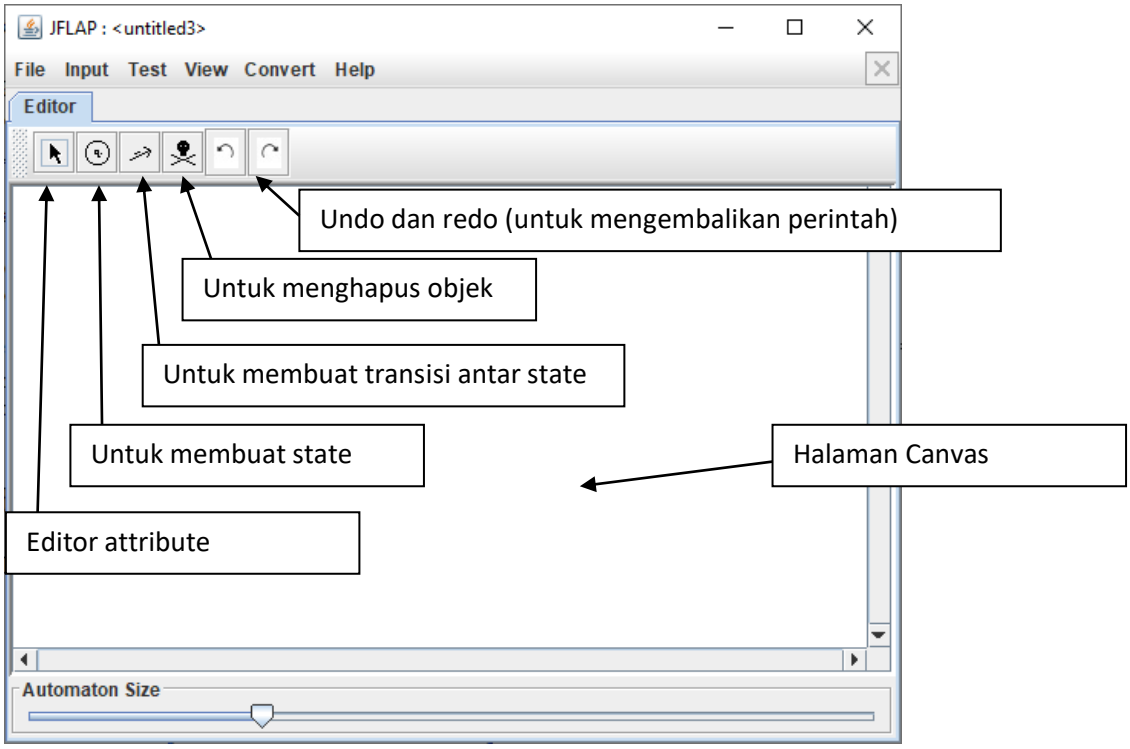
B. Petunjuk Praktikum

1. Lakukan instalasi JFLAP
2. Jalankan JFLAP sehingga muncul halaman Utama JFLAP seperti Nampak pada Gambar 4.1.



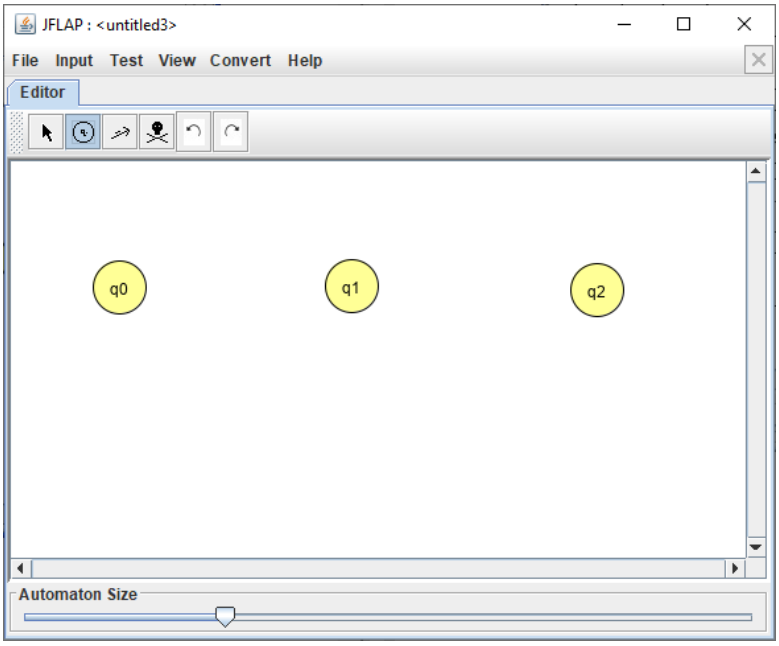
Gambar 4.1. Tampilan rancangan form FSA

3. Pilih Finite Automata, maka akan tampil laman untuk simulasi Finita Automata seperti tampak pada gambar 4.2.



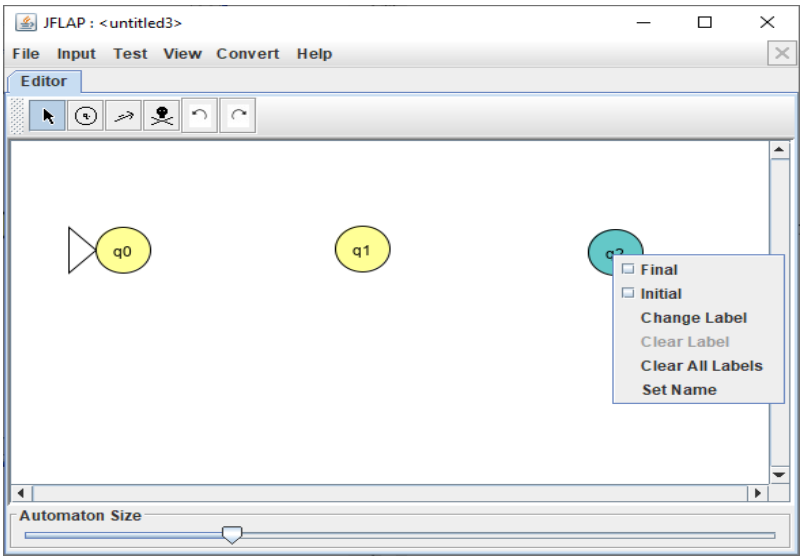
Gambar 4.2. Halaman utama simulasi Finite Automata

4. Membuat State q0, q1, dan q2.



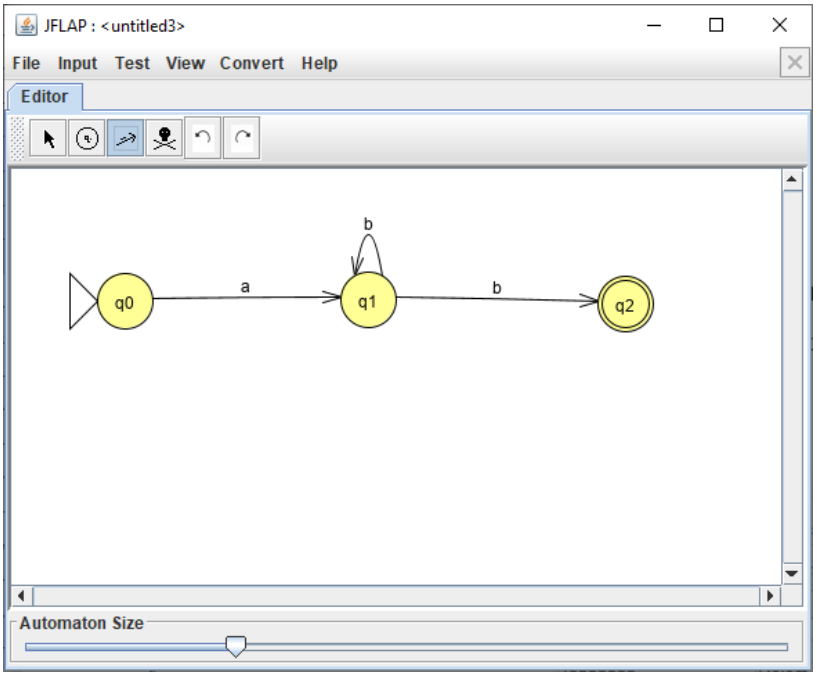
Gambar 4.3. Membuat state q0, q1, dan q2.

5. Menentukan initial state qs, dan final state F atau qf., klik kanan pada q0 dan pilih initial. Selanjutnya, klik kanan pada q3 dan pilih final (Gambar 4.4.).



Gambar 4.4. Menentukan initial state (q0), dan final state (q2).

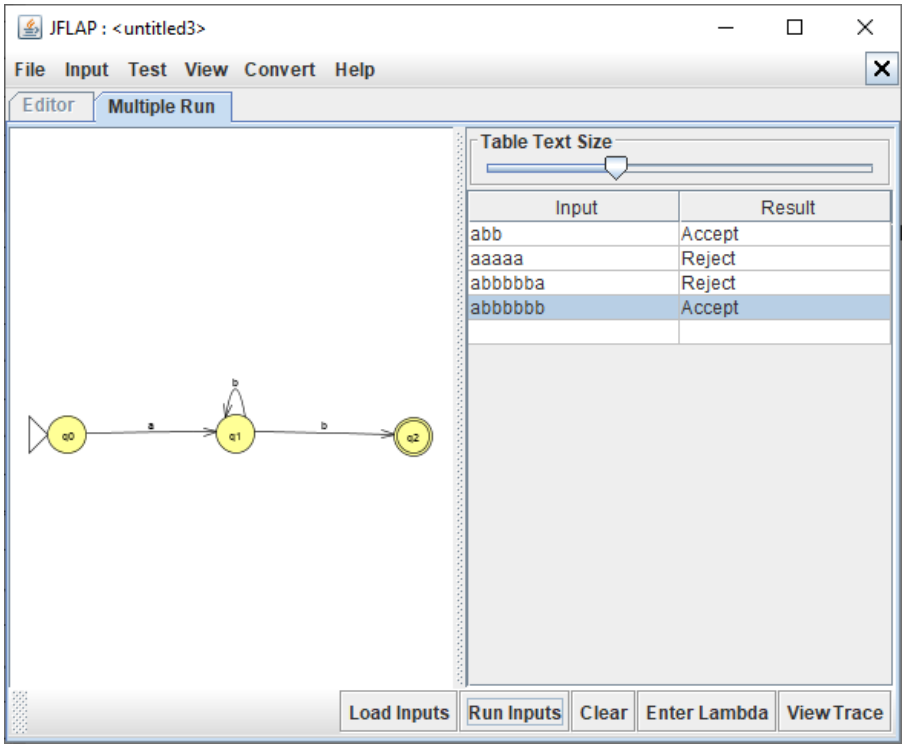
6. Tentukan transisi seperti tampak pada gambar 4.5.



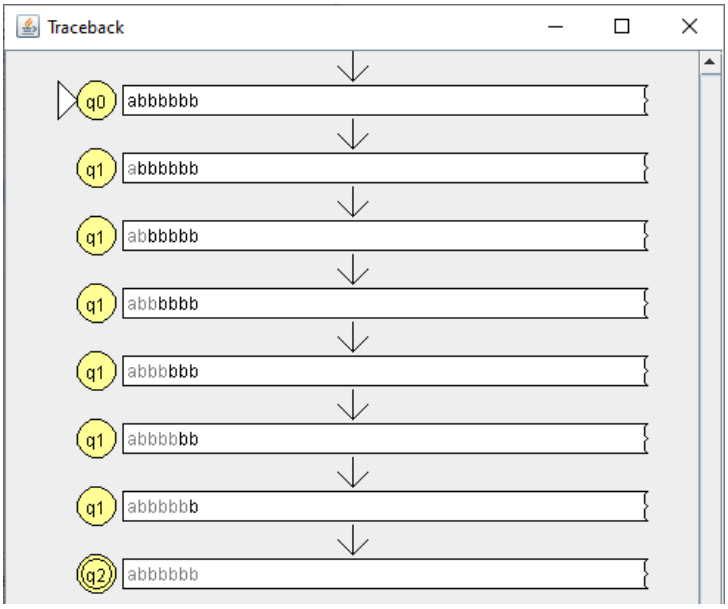
Gambar 4.5. Membuat transisi antar state.

7. Running mesin Finite State Automate tersebut dengan memilih menu **input**, dan **multiple run**. Kemudian inputkan string seperti tampak pada

gambar 4.6. Kemudian, klik **Run Inputs.** Contoh Traceback untuk string "abbbbbb" disajikan pada Gambar 4.7 dengan cara pilih string tersebut pada input, dan klik tombol **View Trace**.



Gambar 4.6. Running input untuk mesin FSA.

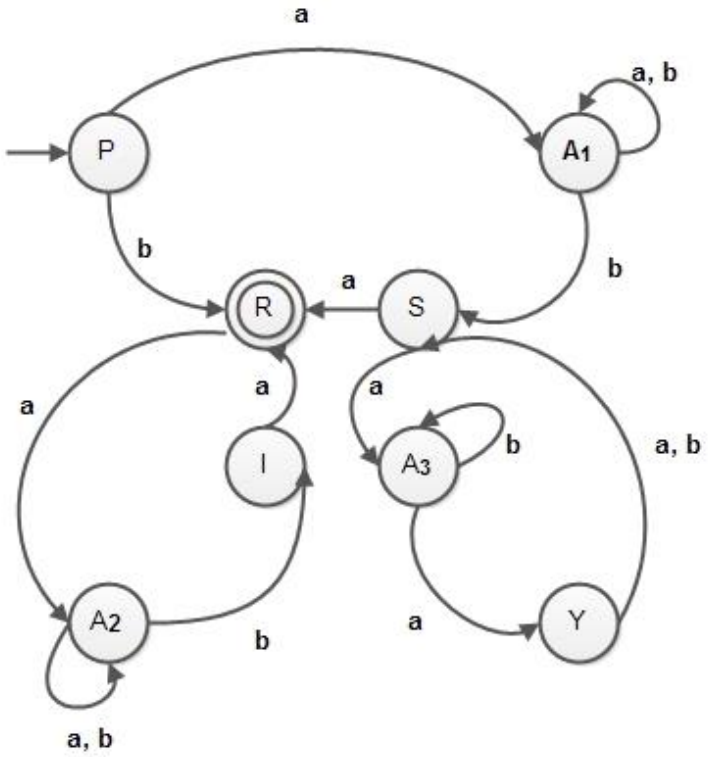


Gambar 4.7. Traceback string "abbbbbb" untuk mesin FSA

C. Tugas Praktikum

Seperti yang anda ketahui di kelas, bahwa Finite stete automate terdiri dari Deterministics finite automata (DFA), dan Non-Deterministics Finite Authomata (NDFA), perhatikan Gambar 4.8, dan tentukan:

1. Q, Σ, δ, qs/S, dan F !
2. Lakukan identifikasi jenis mesin tersebut apakah masuk ke dalam DFA atau NFA! Jelaskan !
3. Buat struktur Mesin tersebut dengan menggunakan JFLAP.
4. Cek L1 = “abababababab “ dan L2 = “bababababa”, apakah diterima mesin tersebut ?



Gambar 4.8. Graph FSA M1

Nilai	Yogyakarta, ……. Paraf asisten <……>

Jawaban Postest

Non-Deterministic Finite Automata (NFA)

Pertemuan	:	V
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* untuk *Non-Deterministic Finite Automata* 2. Mahasiswa mampu memahami algoritma *Non-Deterministic Finite Automata*
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming* 2. Mahasiswa mampu membuat fungsi dan aplikasi untuk *Non-Deterministic Finite Automata*

A. Dasar Teori

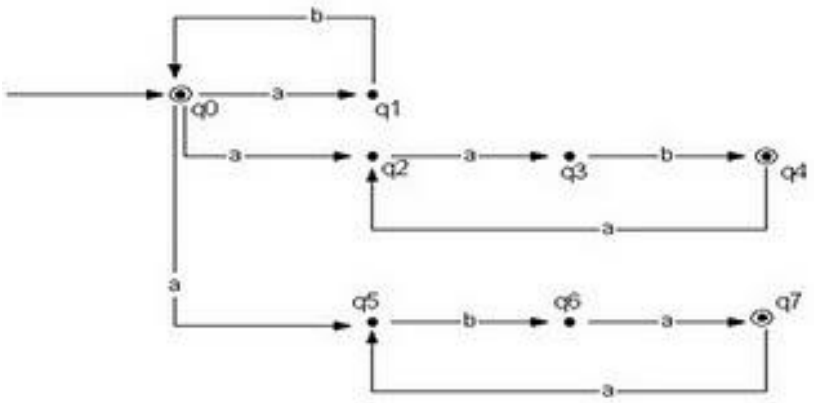
Nondeterministic Finite Automata (NFA) adalah salah satu bagian dari otomata berhingga atau *Finite State Automata (FSA)*. Pada *Nondeterministic Finite Automata (NFA)* dimungkinkan satu simbol menimbulkan transisi ke lebih dari satu kondisi dan memberikan beberapa kemungkinan gerakan sehingga keluarannya tidak dapat dipastikan. Selain itu dimungkinkan juga terjadinya transisi spontan atau *transisi –ε*.

Nondeterministic Finite Automata (NFA) didefenisikan sebagai M yang merupakan sebuah koleksi dari 5 obyek ($\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{s}, \boldsymbol{F}, \Delta$) dimana :

1. $\boldsymbol{Q}$ adalah sebuah himpunan hingga dari kedudukan-kedudukan.
2. Σ adalah sebuah abjad masukan.
3. s adalah salah satu kedudukan di dalam Q yang ditetapkan sebagai kedudukan permulaan.
4. F adalah sebuah koleksi dari kedudukan-kedudukan yang diterima atau final (koleksi / himpunan dari kondisi akhir).

5. Δ adalah sebuah relasi pada (Q x Σ) x Q dan dinamakan relasi transisi.

Salah satu rangkaian *Nondeterministic Finite Automata* (NFA) terlihat pada gambar.

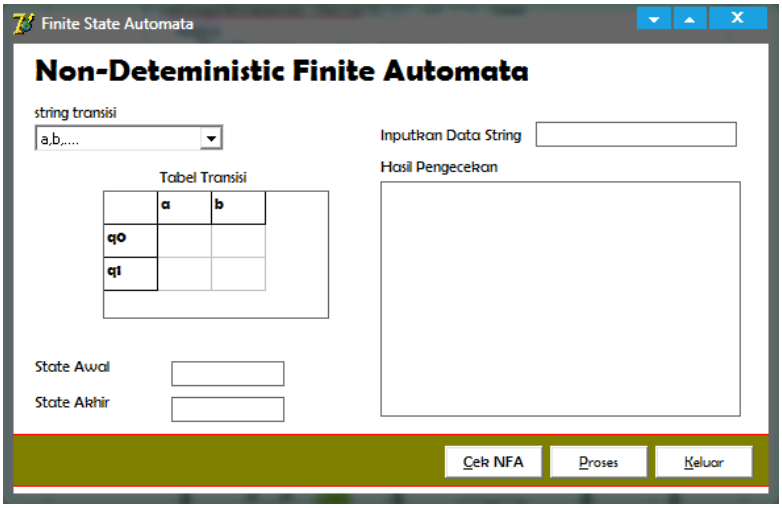


Gambar 5.2. *Rangkaian Nondeterministic Finite Automata (NFA)*

Rangkaian pada Gambar tergolong dalam *Nondeterministic Finite Automata (NFA)* karena beberapa transisi yang berasal dari satu kondisi yaitu kondisi q_0 memiliki inputan yang sama yaitu ‘a’. *Rangkaian* tersebut akan menerima string ab, aab, aabaab, aba, dan abaaba, tetapi tidak akan menerima string abb dan aabb.

B. Petunjuk Praktikum

1. Jalankan *visual programming*
2. Buat desain form seperti praktikum 3 dan 4 (lihat gambar 5.2).



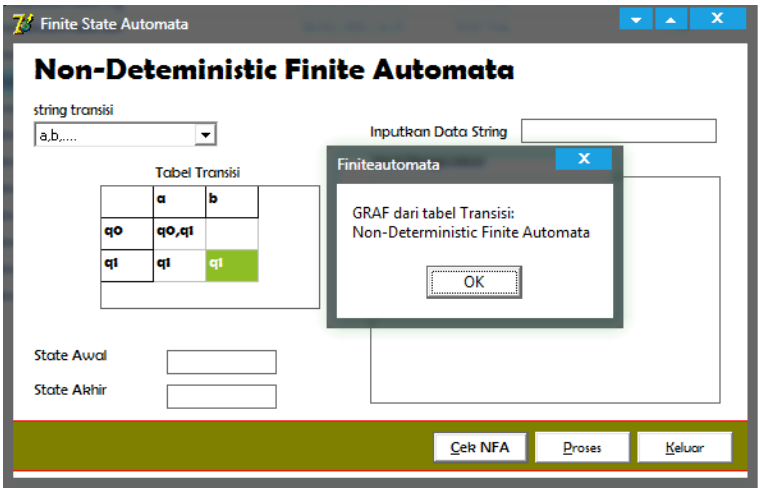
Gambar 5.2 Tampilan form NFA

3. Tambahkan fungsi yang menandakan bahwa graf tersebut NFA dengan aturan

   a. Dalam table transisi ada yang kosong maka dianggap sebagai graf NFA

```
procedure Tfrmutama.cekNFA;
var
  j: Integer;
  i: Integer;
begin
  // TODO -cMM: Tfrmutama.cekDFA default body inserted
  {for i := 1 to strngrdtransisi.RowCount do
  begin
    InputStateA[i] := strngrdtransisi.Cells[1,i];
    InputStateB[i] := strngrdtransisi.Cells[2,i];
  end;  }
  j:=1;

  if (strngrdtransisi.Cells[1,j] <> '') and
(strngrdtransisi.Cells[2,j] <> '') then
  begin
    ShowMessage('tidak ksosong');
    Inc(j);
  end
  else begin
    ShowMessage('GRAF dari tabel Transisi:'+#13+'Non-
Deterministic Finite Automata');
  end;
end;
```

   b. Jika *input* transisi yang sama lebih dari 1 state maka di anggap sebagai NFA



Gambar 5.3 hasil pengecekan graf dari table traansisi

C. Tugas Praktikum

Buat NFA yang menerima string di bawah ini : ab,aab,abb,aaab,abbb. Gunakan NFA dari kasus di atas dan kembangkan untuk mendapatkan string yang ditentukan. Gambar Graf transisinya.

Nilai	Yogyakarta, ……. Paraf asisten <……>
Jawaban Postest	

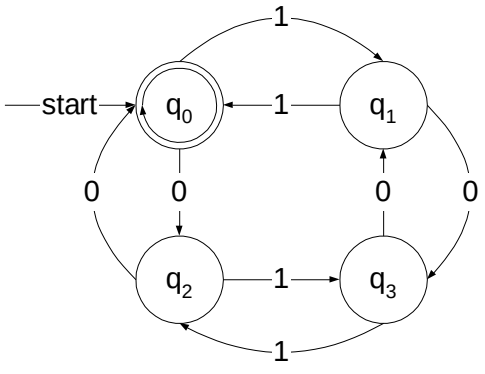
Deterministic Finite Automata (DFA)

Pertemuan	:	VI
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* untuk *Deterministic Finite Automata* 2. Mahasiswa mampu memahami algoritma *Deterministic Finite Automata* (DFA)
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming* 2. Mahasiswa mampu membuat fungsi untuk mengecek DFA dan aplikasi DFA

A. Dasar Teori

Finite state automata berdasarkan kemampuan berubah state-statenya bisa dikelompokkan ke dalam *deterministic* maupun *non-deterministic*.

DFA mempunyai cirri utama yaitu dari satu *state* tepat satu *state* berikutnya untuk setiap simbol masukkan yang diterima. Lihat gambar di bawah



$Q = \{q_0 , q_1 , q_2 , q_3 \}$

$\Sigma = \{0,1\}$

$S = q_0$

$F = \{ q_0\}$

fungsi transisi

δ	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Jika ada string

0011 : diterima.

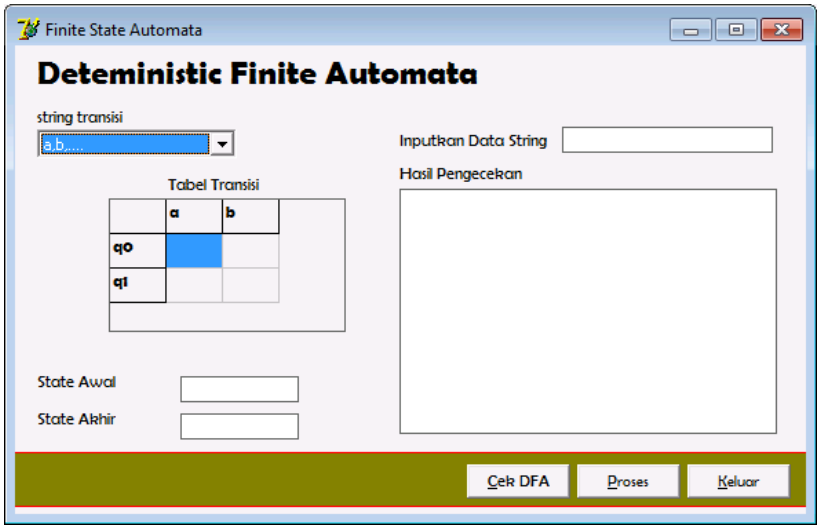
10010 : ditolak, karena banyaknya 0 ganjil

δ(q0,011)= δ(q2,11) =δ(q3,1)= q2 Ditolak

δ(q0,1010)= δ(q1,010) =δ(q3,10)=δ(q2,0)= q0 Diterima

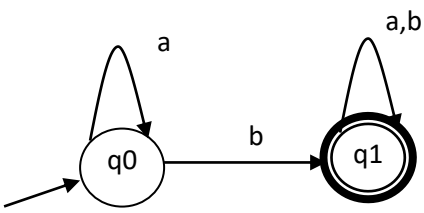
B. Petunjuk Praktikum

1. Jalankan program *visual programming*
2. Buat program seperti pada *finite state automata*



Gambar 6.1. tampilan form *Deterministic Finite Automata*

3. Gunakan Diagram DFA di bawah ini untuk membuat program

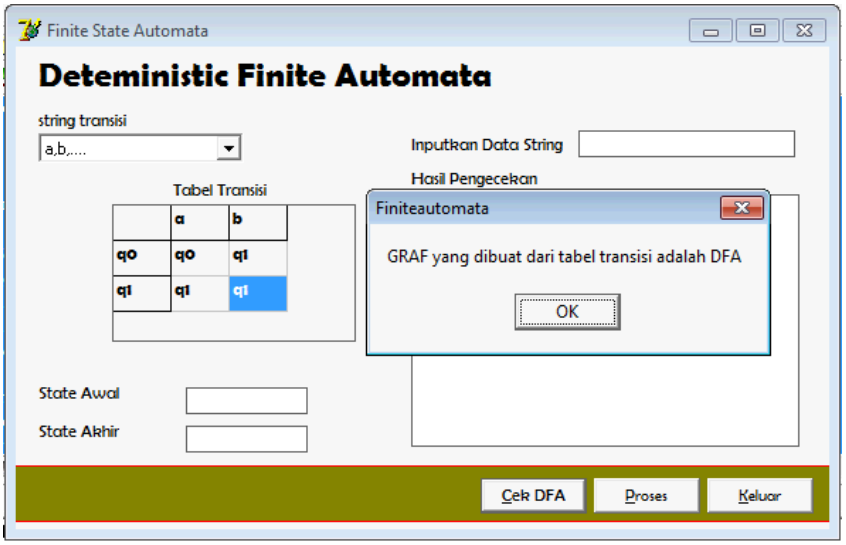


Tabel Transisi

	a	b
q0	q0	q0
q1	q1	q1

4. Tambahkan fungsi untuk pengecekan bahwa graf DFA

```
procedure Tfrmutama.cekDFA;
var
  j: Integer;
  i: Integer;
begin
  for j := 1 to strngrdtransisi.RowCount - 1  do
  begin
    if (strngrdtransisi.Cells[1,j] = '') and
(strngrdtransisi.Cells[2,j] = '')  then
    begin
      ShowMessage('bukan GRAF DFA, karena ada input yang kosong');
    end
    else begin
      ShowMessage('GRAF yang dibuat dari tabel transisi adalah
DFA');
    end;
  end;
end;
```



Gambar 6.2. Tampilan form DFA cek graf

C. Tugas Praktikum

Tambahkan 1 state dengan nama q2. Dan table transisinya

	A	B
q0	q0	q0
q1	q2	q1
q2	q2	q2

Gambar graf transisinya dan tentukan 2 string yang diterima dan ditolak (tuliskan hasilnya pada lembar jawaban di modul)

Nilai	Yogyakarta, ……. Paraf asisten <……>
Jawaban Postest	

Pushdown Automata (PDA)

Pertemuan	:	VII
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu mensimulasikan Pushdown Automata (PDA), khususnya non-deterministik PDA dengan menggunakan JFLAP. 2. Mahasiswa mampu memahami non-deterministik PDA (NDFA)
Indikator	:	1. Mahasiswa mampu membuat struktur NDFA dan mensimulasikannya. 2. Mahasiswa mampu menjelaskan proses NDFA.

A. Dasar Teori

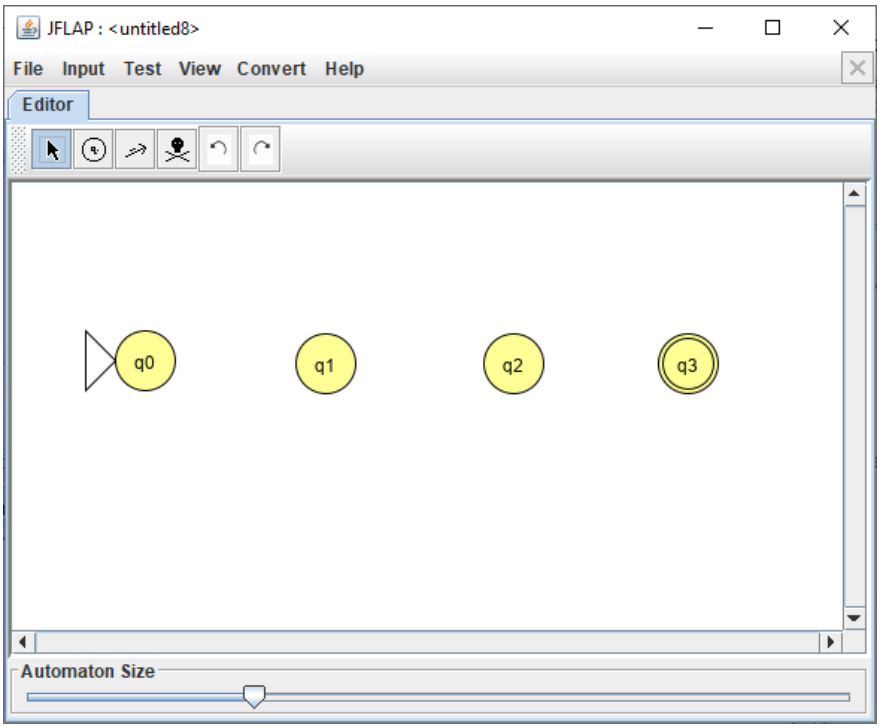
Pushdown Automata (PDA) adalah sebuah finite automata yang memiliki memori berdasarkan *stack* (tumpukan). Di dalam sebuah PDA setiap transisi didasarkan pada input symbol yang terbaca terakhir, dan juga input symbol yang berada pada tumpukan (stack) teratas. Operasi stack ini terdiri dari pop dan push. Pop mengeluarkan input symbol dari tumpukan, dan push memasukkan symbol ke dalam tumpukan. Sebagai initial, di dalam tumpukan terdapat symbol Z0 yang mengindikasikan tumpukan paling bawah.

Di dalam JFLAP, sebuah nondeterministic pushdown automata (NPDA) M didefinnisikan menggunakan 7 tupel yaitu M = (Q, Σ, Γ, δ, qs, Z, F) dimana,
Q merupakan sebuah himpunan finite state {qi | i adalah nonnegative integer}
Σ merupakan input symbol, alphabet.
Γ adalah finite stack alphabet.
δ adalah fungsi transisi, $\delta : Q \times \Sigma^* \times \Gamma^* \rightarrow$ finite subsets of $Q \times \Gamma^*$
qs (anggota Q) yang merupakan initial state.
Z adalah symbol untuk stack awal (harus ditulis dengan kapital Z)

F (anggota himpunan Q) adalah himpunan final state

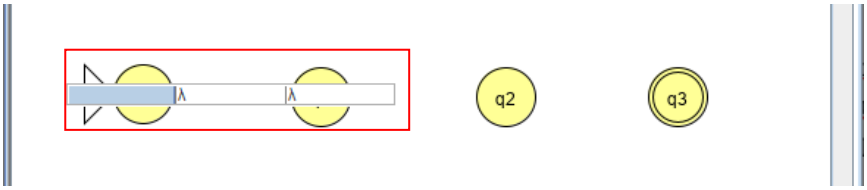
B. Petunjuk Praktikum

1. Jalankan aplikasi JFLAP, dan pilih menu “Pushdown Automata”.
2. Buat State q0 sd q3, tentukan q0 sebagai initial state dan q3 sebagai final state (Gambar 7.1). Input PDA berbeda dengan kita menentukan input pada finite automata. Pada PDA kita memerlukan 3 buah input dalam setiap transisi (Gambar 7.2).



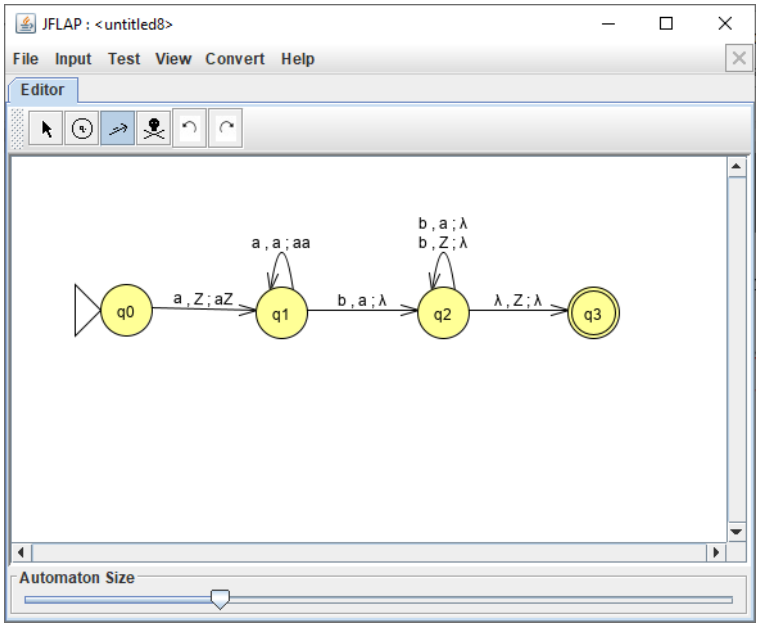
Gambar 7.1. Menentukan state.

Nilai input pertama dalam setiap transisi menunjukkan input yang diproses. Nilai berikutnya menunjukkan nilai pada top stack, dan nilai ketiga pada sebuah transisi menunjukkan nilai baru yang dipush menjadi top stack ketika transisi setelah mem-pop nilai yang berada pada top stack.



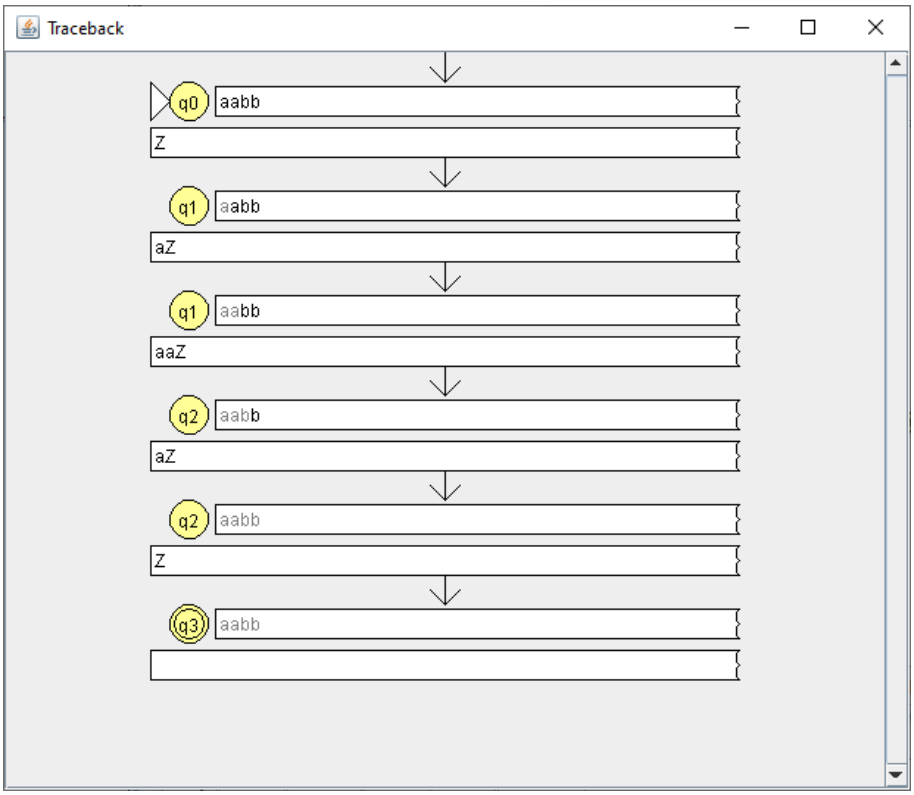
Gambar 7.2. Input pada PDA

3. Kontruksipak PDA seperti tampak pada Gambar 7.3.



Gambar 7.3. Kontruksi PDA (m1)

4. Masukkan input L=“aabb”.
5. Hasil ketika dilakukan trace tampak pada Gambar 7.4.



Gambar 7.4. Hasil Trace string “aabb”

C. Tugas Praktikum

1. Tentukan masing-masing 5 buah string atau L yang diterima dan ditolak oleh mesin m1

2. Kontruksikan PDA dari tabel transisi M2 sebagai berikut.

   Q={q0, q1 }

   ∑={a,b}

   Γ ={A,B,Z}

   qs=q0

   Z=Z

   F={q1}

   δ (Fungsi transisi):

   δ(q0,ε,Z)={(q1,Z)}

   δ (q0,a,Z)={(q0,AZ)}

   δ (q0,b,Z)={(q0,BZ)}

   δ (q0,a,A)={(q0,AA)}

   δ (q0,b,A)={(q0, ε)}

   δ (q0,a,B)={(q0, ε)}

   δ (q0,b,B)={(q0, BB)}

   Tentukan 5 string atau L yang diterima oleh M2 tersebut.

Nilai	Yogyakarta, …… Paraf asisten <……>
Jawaban Postest	

Chomsky Normal Form (CNF)

Pertemuan	:	VIII
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* CNF 2. Mahasiswa mampu memahami algoritama d dari proses *CNF*
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming* 2. Mahasiswa mampu membuat fungsi untuk CNF

A. Dasar Teori

Salah satu bentuk yang normal yang berguna untuk tata bahasa bebas konteks/context free grammar (CFG) adalah Bentuk Normal Chomsky/Chomsky Normal Form (CNF). CNF dari sebuah CFG dapat dibentuk setelah CFG disederhanakan, yaitu setelah CFG tidak memiliki produksi Useless, Unit, dan Nullable (ε). CFG telah dalam bentuk CNF apabila sisi kanan (β) dari sebuah CNF berisi (Cole, 2007):

a) Maksimal sebuah terminal, contoh A → b, atau
b) Maksimal terdiri dari dua buah variable A → BC, atau
c) Khusus pada aturan produksi symbol awal S → ε , jika ε di dihasilkan oleh produksi S (dalam bahasa),
d) Simbol awal S hanya muncul pada sisi kiri (α) aturan produksi.

Contoh bentuk normal Chomsky:

S → CD
C→ AB | BD
A→ a
B →b
D → d

B. Petunjuk Praktikum

1. Jalankan *visual programming* atau bahasa pemrograman lain yang dikuasai
2. Buat desain form input dan luaran untuk proses pembentukan normal Chomsky
3. Buat prosedur dan fungsi untuk membuat normal Chomsky berdasarkan algoritma sebagai berikut (Branicky, 2004):

1. Untuk setiap aturan produksi A → B_1B_2 .. B_k yang berisi sebuah terminal σ
   1) Tambahkan symbol variable baru (non terminal) dan bentuk aturan produksi baru menjadi Cσ → σ (kecuali aturan produksi tersebut telah tersedia)
   2) Ganti setiap B_i = σ dengan Cσ
2. Ubah setiap aturan produksi A → B_1B_2 … B_k dengan B_1 .. B_k berupa symbol variable dan k ≥ 3, maka lakukan
   1) $A \rightarrow B_1A_1$
   2) $A_1 \rightarrow B_2A_2$
   3) $A_2 \rightarrow B_3A_3$
   4) ….
   5) $A_{k-2} \rightarrow B_{k-1}B_k$ dimana A_i adalah variable baru
3. Gabungkan hasil pada langkah 1 dan 2

4. Lakukan testing untuk memastikan program berjalan dengan benar
5. Dokumentasikan form/interface, prosedur, fungsi, input dan luaran program yang dibuat dalam kertas A4, diketik dengan huruf times new roman 12 dengan 1 spasi.

C. Pretest Praktikum

Buatlah bentuk CNF dari CFG sebagai berikut menggunakan program yang telah anda buat.

S → AB | Ba

A → dB | b

B → Bb | a

C → c

D. Postest Praktikum

Gunakan program yang anda buat untuk membuat CNF dari CFG berikut:

S → ABCD | BCd | CD

A → bB | Cd

B → cD | b

C → c

D → aBc | d

E. Referensi

Branicky, M. (2004). EECS 343 Handout: Chomsky Normal Form. Retrieved from http://dora.eeap.cwru.edu/msb/343/normal.pdf

Cole, R. (2007). *Converting CFGs to CNF (Chomsky Normal Form)*. Retrieved from http://www.cs.nyu.edu/courses/fall07/V22.0453-001/cnf.pdf

Nilai	Yogyakarta, ……. Paraf asisten <……>
Jawaban Postest	

Penghilangan Rekursif Kiri

Pertemuan	:	IX
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* penghilangan rekursif kiri 2. Mahasiswa mampu memahami algoritama d dari proses *penghilangan rekursif kiri*
Indikator	:	1. Mahasiswa mampu membuat *interface* dengan menggunakan *visual programming atau bahasa pemrograman lain yang dikuasai* 2. Mahasiswa mampu membuat prosedur/fungsi untuk penghilangan rekursif kiri

A. Dasar Teori

Conext Free Grammar (CFG) dikatakan rekursif kiri apabila aturan produksinya dalam bentuk (Maza, 2004):

$$A \rightarrow A\beta$$

Dimana A merupakan symbol variable (non terminal) dan β adalah string dari symbol-simbol CFG. Rekursif kiri akan mengakibatkan perulangan penurunan CFG yang tidak akan pernah berhenti (Weimar et al., 2015). Oleh karena itu, rekursif kiri dalam CFG harus dihilangkan.

Contoh aturan produksi rekursif kiri:

S→ SdBa
A → AbCD
C → Cda
D → DbacBa

Langkah-langkah untuk penghilangan rekursif kiri (Moore, 2001; Utdirartamo, 2005):

1. Pisahkan aturan produksi yang rekursif kiri dan yang tidak, missal:
   a. Aturan produksi yang rekursif kiri $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \ldots \mid A\alpha_n$, sehingga dapat diketahui $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$
   b. Aturan produksi yang tidak rekursif kiri $A \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \ldots \mid \beta_m$, sehingga dapat diketahui $\beta_1, \beta_2, \beta_3, \ldots, \beta_m$

2. Lakukan penggantian aturan produksi yang terdapat rekursif kiri berdasarkan rumus sebagai berikut:

   $A \rightarrow \beta_1 \mid \beta_1 A' \mid \beta_2 \mid \beta_2 A' \mid \beta_3 \mid \beta_3 A' \mid \ldots \mid \beta_m \mid \beta_m A'$

   $A' \rightarrow \alpha_1 \mid \alpha_1 A' \mid \alpha_2 \mid \alpha_2 A' \mid \alpha_3 \mid \alpha_1 A' \mid \ldots \mid \alpha_n \mid \alpha_n A'$

   Dimana A' marupakan symbol variable (non terminal) yang tidak digunakan di aturan produksi lain dalam CFG tersebut.

B. Petunjuk Praktikum

1. Jalankan *visual programming* atau bahasa pemrograman lain yang dikuasai
2. Buat desain form input dan luaran untuk proses penghilangan rekursif kiri.
3. Buat prosedur/fungsi untuk proses penghilangan rekursif kiri berdasarkan algoritma sebagai berikut:

1. Pisahkan produksi yang rekursif kiri dan tidak
   1) Produksi rekursif kiri $A\alpha_1, A\alpha_2, A\alpha_3, \ldots, A\alpha_n$
   2) Produksi tidak rekursif kiri $\beta_1, \beta_2, \beta_3, \ldots, \beta_m$
2. Ganti setiap produksi dalam bentuk $Ai \rightarrow Aj\alpha$ dengan produksi $Ai \rightarrow \beta_1 \mid \beta_1 A' \mid \beta_2 \mid \beta_2 A' \mid \beta_3 \mid \beta_3 A' \mid \ldots \mid \beta_m \mid \beta_m A'$, Dan $A' \rightarrow \alpha_1 \mid \alpha_1 A' \mid \alpha_2 \mid \alpha_2 A' \mid \alpha_3 \mid \alpha_1 A' \mid \ldots \mid \alpha_n \mid \alpha_n A'$
3. Lakukan langkah 1 dan 2 untuk semua aturan produksi pada CFG

4. Lakukan testing untuk memastikan program berjalan dengan benar
5. Dokumentasikan form/interface, prosedur, fungsi, input dan luaran program yang dibuat dalam kertas A4, diketik dengan huruf times new roman 12 dengan 1 spasi.

C. Pretest Praktikum

Buatlah bentuk CNF dari CFG sebagai berikut menggunakan program yang telah anda buat.

S → SABC | Ba | ab | a | c

D. Postest Praktikum

Gunakan program yang anda buat untuk membuat CNF pada kasus CFG berikut:

S → Sba | Abc | b

A → AbB | ab | Ac | d

B → BcA | b

E. Referensi

Maza, M. M. (2004). Elimination of left recursion. Retrieved from http://www.csd.uwo.ca/~moreno//CS447/Lectures/Syntax.html/node8.html

Moore, R. C. (2001). Removing Left Recursion from Context-Free Grammars. Revised version of paper appearing in Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics, ANLP-NAACL 2000.

Utdirartamo, F. (2005). *Teori Bahasa dan Otomata*. Yogyakarta: Penerbit Graha Ilmu.

Weimar, W., Leach, K., Thomason, Wi., Recto, R., Roberts, J., & Leath, H. (2015). Eliminating Immediate Left Recursion. Retrieved from http://www.cs.virginia.edu/~cs415/reading/RemovingLeftRecursion.pdf

Nilai	**Yogyakarta, …….** **Paraf asisten** **<……>**

Jawaban Postest

Greibach Noral Form (GNF)

Pertemuan	:	X
Alokasi Waktu	:	1,5 jam
Kompetensi Dasar	:	1. Mahasiswa mampu membuat rancangan *interface* Greibach Normal Form (GNF) 2. Mahasiswa mampu memahami algoritama dari proses *GNF*
Indikator	:	1. Mahasiswa mampu membuat *interface GNF* dengan menggunakan *visual programming atau bahasa pemrograman lain yang dikuasai* 2. Mahasiswa mampu membuat prosedur/fungsi untuk GNF

A. Dasar Teori

Greibach Normal Form (GNF) merupakan bentuk normal yang memiliki banyak konsekuensi teoritis dan praktis (Utdirartamo, 2005). Sebuah CFG dikatakan mememuhi bentuk normal greibach (GNF) apabila setiap produksi memenuhi bentuk (Gautam, 2011).

$$A \rightarrow a\alpha, \text{ dimana } a \in T \text{ dan } \alpha \in V^*$$

Sehingga dapat disumpulkan bahwa CFG yang memeneuhi GNF pada ruas kanan (β) diawali oleh maksimum satu symbol terminal dan selanjutnya berisi symbol-simbol variable, atau hanya terdiri dari sebuah symbol terminal saja. Agar CFG dapat diubah ke GNF, maka harus memenuhi syarat, yaitu (Utdirartamo, 2005) :

1. Sudah dalam betuk normal Chomsky
2. TIdak bersifat rekursif kiri
3. Tidak menghasilkan ε

Contoh aturan produksi CFG yang sudah memenuhi GNF:

$S \rightarrow aBC \mid b$
$B \rightarrow aC \mid aB \mid b$
$C \rightarrow cBDB \mid b$

B. Petunjuk Praktikum

1. Jalankan *visual programming* atau bahasa pemrograman lain yang dikuasai
2. Buat desain form input dan luaran untuk proses pembentukan GNF
3. Buat prosedur/fungsi untuk pembentukan GNF berdasarkan algoritma sebagai berikut:

1. Ubah nama semua symbol variable menjadi urutan A_1, A_2, A_3, … A_n, dimana A_1= S (symbol awal)
2. Seluruh aturan produksi yang ruas kanannya diawali dengan symbol Variabel, dapat dituliskan dalam bentuk $A_h \rightarrow A_i\gamma$ dimana h<>i (Rekursif kiri sudah dihilangkan), γ berupa symbol variable.
3. JIka h<i, aturan produksi ini sudah benar (tidak perlu diubah)
4. JIka h>i, aturan produksi belum benar. Lakukan substitusi berulang-ulang terhadap A_i (ganti A_i pada produksi ini dengan ruas kanan produksi dari variable A_i)sampai diperolah aturan produksi dalam bentuk $A_h \rightarrow A_p\gamma$ (dimana $h \leq p$), dimana :
   1) Jika h=p, lakukan penghilangan rekursif kiri
   2) JIka h<p, aturan produksi sudah bener
5. Jika terjadi pengilangan rekursif kiri pada tahap 4, sejumlah symbol variable yang muncul dari operasi ini dapat disisipkan pada urutan variable semula di mana saja asalkan tidak ditempat sebelum A_h.
6. Setelah langkah 4 dan 5 dilakkan, maka aturan-aturan produksi yang ruas kanannya dimulai

simbnol variable sudah berada dalam bentuk urutan yang benar.

$A_x \rightarrow A_y\gamma$ (dimana x <y)

Produksi-produksi yang lain ada dalam bentuk:

$A_x \rightarrow a\gamma$ (a=symbol terminal)

$B_x \rightarrow \gamma$

(B_x = symbol variable yang muncul akibat operasi penghilangan rekursif kiri

7. Lakukan substitusi mundur mulai dari variable An, An-1, An-2,… Dengan cara ini aturan produksi dalam bentuk Ax → $A_y\gamma$ dan $B_x \rightarrow \gamma$ dapat diubah, sehingga ruas kanannya dimulai dengan symbol Terminal

4. Lakukan testing untuk memastikan program berjalan dengan benar
5. Dokumentasikan form/interface, prosedur, fungsi, input dan luaran program yang dibuat dalam kertas A4, diketik dengan huruf times new roman 12 dengan 1 spasi.

C. Pretest Praktikum

Buatlah bentuk GNF dari CFG sebagai berikut (Utdirartamo, 2005) menggunakan program yang telah anda buat.

S → CA

A → a | d

B → b

C → DD

A → AB

D. Postest Praktikum

Gunakan program yang anda buat untuk membuat CNF pada kasus CFG berikut:

S → BC

A → DB | b

B → CA | a

C → AC | c

D → d

E. Referensi

Gautam, K. Das. (2011). Properties of Context-free Languages. Retrieved from http://www.iitg.ernet.in/gkd/ma513/oct/oct18/note.pdf

Utdirartamo, F. (2005). *Teori Bahasa dan Otomata*. Yogyakarta: Penerbit Graha Ilmu.

Nilai	**Yogyakarta, …….** **Paraf asisten** **<……>**

Jawaban Postest